Wanderwege

Die Organisation „Wandern in Feld und Flur (WFF)" kümmert sich seit vielen Jahren um die Belange von Wanderern. In diesem Zusammenhang bietet sie eine Software mit vorgegebenen Wanderwegen an. In Zukunft möchte WFF den Nutzern die Möglichkeit geben, eigene Wanderwege zusammenzustellen. Aus diesem Grund entwickelt sie die Software weiter, so dass die Nutzer Wegabschnitte zu neuen Wanderwegen zusammensetzen können.

In der bisherigen Software der Organisation sind die Wanderwege als Felder von Stationen gespeichert. Jede Station hat einen Namen, der sie eindeutig identifiziert, und an jeder Station kann eine Schutzhütte stehen.

In der neuen Software sollen die Wanderwege einen Namen haben und als Listen von Abschnitten implementiert werden. Ein Abschnitt führt jeweils von einer Station zu einer weiteren (evtl. derselben) Station und zeichnet sich durch seine Länge in km und seinen Schwierigkeitsgrad (1, 2 oder 3) aus. Aus diesen beiden Werten lässt sich eine gute Näherung für die Dauer der Wanderung auf diesem Abschnitt berechnen. Bei Schwierigkeitsgrad 1 geht man von einer Durchschnittsgeschwindigkeit von 4,2 km/h aus, pro weiterem Schwierigkeitsgrad sinkt diese um jeweils 0,5 km/h.

Aufgaben

1 Erläutern Sie, weshalb es im beschriebenen Kontext sinnvoller ist, einen Wanderweg als Liste (von Abschnitten) statt als Feld (von Stationen) zu implementieren.

(4 BE)

2 Modellieren Sie unter Berücksichtigung des Einführungstextes und aller folgenden Aufgaben die Klassen *TWanderweg*, *TAbschnitt* und *TStation* mit deren Beziehungen als UML-Klassendiagramm für die neue Software. Auf die Angabe der get/set-Methoden kann verzichtet werden.

(6 BE)

3 Um die Wanderwege aus der alten Software in die neue zu überführen, muss aus einem Feld von Stationen eine Liste von Abschnitten generiert werden.

Implementieren Sie die Methode *erzeugeAbschnitte(Stationen: TStationenArray)* der Klasse *TWanderweg*, mit der aus dem Feld *Stationen* die Abschnittsliste erzeugt wird und die den ersten Abschnitt im Attribut *ersterAbschnitt* der Klasse *TWanderweg* ablegt. Es kann davon ausgegangen werden, dass nur gültige Wanderwege übergeben werden.

Hinweis
Die Werte für die Attribute *Laenge* und *Schwierigkeitsgrad* eines Abschnitts können ignoriert werden.

(5 BE)

4.1 Implementieren Sie die Methode *berechneDauer* der Klasse *TAbschnitt*, die die Dauer der Wanderung auf diesem Abschnitt berechnet und zurückgibt.

Hinweise
Die Werte der Attribute *Laenge* und *Schwierigkeitsgrad* sind dabei korrekt belegt.
Die Dauer berechnet sich als Quotient von Länge und Durchschnittsgeschwindigkeit.

(2 BE)

4.2 Entwerfen Sie ein Struktogramm für einen Algorithmus, der die Gesamtdauer einer Wanderung auf einem Wanderweg berechnet.

(2 BE)

5 Fernwanderer sind an einem möglichst schnellen Vorankommen interessiert. Sie möchten deshalb, dass ihr Wanderweg keinen Rundweg als Bestandteil enthält.

Analysieren Sie die beiden Methoden *ermittleRundweg* und *sucheRundweg(...)* der Klasse *TWanderweg* (Material 1).

(9 BE)

6.1 In Material 2 befindet sich eine graphische Darstellung eines Wanderweges mit der Abschnittsfolge A-B-C-D-E-F-G, der einen Rundweg besitzt. Geben Sie die Abschnittsfolge des Wanderweges nach Entfernung des Rundweges an und erläutern Sie ihre Angabe.

(3 BE)

6.2 Beschreiben Sie einen Algorithmus, der ermittelte Rundwege aus einem beliebigen Wanderweg entfernt.

(4 BE)

Material 1

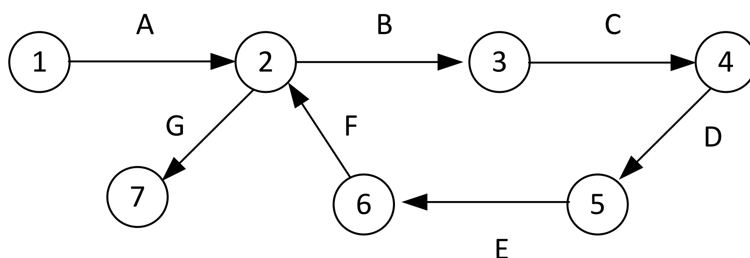
Die Methoden *ermittleRundweg* und *sucheRundweg(…)* der Klasse *TWanderweg*

```
01 procedure TWanderweg.ermittleRundweg;
02 begin
03   if not sucheRundweg(ersterAbschnitt) then
04     schreibe('Kein Rundweg gefunden!');
05 end;
06
07 function TWanderweg.sucheRundweg(einAbschnitt: TAbschnitt): boolean;
08 var gefunden: boolean;
09     laufAbschnitt: TAbschnitt;
10     Beginn, Ende: String;
11 begin
12   gefunden:= false;
13   if einAbschnitt.getNaechsterAbschnitt <> nil then
14     gefunden:= sucheRundweg(einAbschnitt.getNaechsterAbschnitt);
15   laufAbschnitt:= einAbschnitt;
16   Beginn:= einAbschnitt.getBeginn.getName;
17   while laufAbschnitt <> nil do begin
18     Ende:= laufAbschnitt.getEnde.getName;
19     if Ende = Beginn then begin
20       schreibe('Rundweg Beginn und Ende: ' + Beginn );
21       gefunden:= true;
22     end;
23     laufAbschnitt:= laufAbschnitt.getNaechsterAbschnitt;
24   end;
25   result:= gefunden;
26 end;
```

Hinweis
Die Methode *schreibe* (Zeile 4 und 20) kann als implementiert vorausgesetzt werden.

Material 2

Graphische Darstellung eines Wanderweges mit einem Rundweg



Hinweis
Die Ziffern 1 bis 7 stellen unterschiedliche Stationen dar.